# 文档目录

文档目录 1

变更记录 1

一、文档功能 1

二、类说明 2

三、设计层次图 2

1. 第一层:CoreData层 2

2. 第二层：WACacheTableHandler层 2

3. 第三层，WACacheTableview 3

4. 第四层，WASearchViewController 4

四、使用方法 4

1、引入框架 4

2、引入头文件，实现基类协议 4

3、初始化 5

a 关键点：CWASearchListViewStyle自定义界面 6

4、实现列表的搜索，上拉刷新，下拉加载的数据请求 7

a 联网请求数据，基类通过以下通知子类请求数据 7

b 子类请求数据，获取到数据后，返回给基类。 7

5、初始化列表cell 8

a 关键点：cell的一些默认属性 9

6、其他的一些委托方法 9

五、基类的一些关键方法 10

## 变更记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变更日期 | 功能点 | 变更内容 | 提出人 | 变更人 |
| 2012-11-5 |  | 重新整理文档 |  | 杨锐 |
| 2012-12-27 |  | 重新整理文档 |  | 杨锐 |

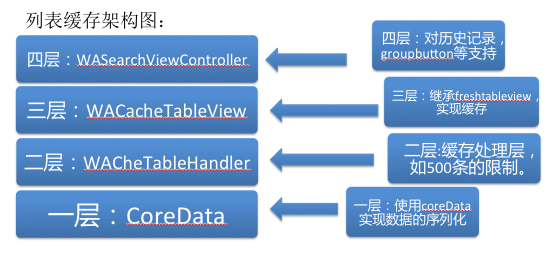
# 一、文档功能

该文档为使用者使用CWASearchViewController而编写，请仔细参照文档。更多信息请参考类的头文件，头文件里面有详尽的说明；也请参考测试工程ListCacheTest

# 二、类说明

CWASearchViewController为一个viewcontroller的继承类，实现了对搜索历史记录的支持，搜索条件condition的动态获取，上拉刷新，下拉加载，数据缓存的支持，groupbutton动态使用，底部区域扩展的支持等功能。

# 三、设计层次图

****

每一层只完成本层的功能，供上层调用，或者调用下层的代码。

实际业务处理时的调用图，如下

### 1. 第一层:CoreData层

该层属于系统层，具体使用请参照系统的coredata使用方法。

### 2. 第二层：WACacheTableHandler层

该层封装了coredata以持久化数据，提供接口供上层调用。

该层把上层调用传入的数据，插入到coredata中，或者删除掉超过范围的数据，关注下面的关键方法。

关键方法：

//初始化函数，aCacheFileName：为缓存的文件名

-(id)initCacheFileName:(NSString\*)aCacheFileName;

/\*!

@method

@abstract 插入objects

@param objectsInsert:要插入的数据，可以以NSMutableArray形式传入

@param aRange:插入的位置,以及数目，数目和objectsInsert数目一致。

@result 返回值是TWASuppliesFromType类型

\*/

- (TWAInsertResultType)insertNewObject:(NSMutableArray\*)objectsInsert

Range:(WAListCacheRange)aRange;

/\*!

@method

@abstract 删除指定范围内的数据

@param aRange:删除的范围

@result 返回TWADeleteResultType

\*/

-(TWADeleteResultType)deletaObjectsAtRange:(WAListCacheRange)aRange;

### 3. 第三层，WACacheTableview

该层继承CWAPullingRefreshTableView，使一个tableview支持上拉下拉刷新，加载。并调用第二层的接口，使tableview支持数据缓存。

该层对于数据缓存来说，没有太多关键方法，都只是调用第二层的接口。

关键函数：

/\*!

@method

@abstract 初始化

@param frame:尺寸大小

@param aPullingDelegate:委托对象

\*/

- (id)initWithFrame:(CGRect)frame

pullingDelegate:(id<PullingRefreshTableViewDelegate>)aPullingDelegate

cacheFileName:(NSString\*)aCacheFileName;

注意的是，在CWAPullingRefreshTableView有些关键代码，具体如下

//消息提示

* (void)flashMessage:(NSString \*)msg

//列表拖动过程中，列表本身状态的变化

- (void)tableViewDidScroll:(UIScrollView \*)scrollView

//添加Cell，当数据加入后，需要增加tableview上的cell

### 4. 第四层，WASearchViewController

该层使用第三层，并支持，历史记录，groupbutton等功能。

关键函数：

请参考本文档的，三，四，五部分。

特别注意的是：

//设置界面，布局界面

-(void)setViewStyle；

//成功返回数据列表数据，添加数据

-(void)addSearchResultWithArray:(NSMutableArray\*)aResultArray；

-(void)addCells:(WAListCacheRange)aAddRange

//准备数据，因为，内存中的Array是一个动态变化的过程。

//当tableview在动态拖动过程中，需要准备将要显示的数据。

-(void)readyForData:(NSIndexPath\*)indexpath

//键盘将要出现，或隐藏，此处会根据系统键盘的frame去调整历史记录列表的 frame，

//使历史记录不会被系统键盘挡住

* (void)keyboardWillShow:(NSNotification \*)n

- (void)keyboardWillHidden:(NSNotification\*)n

# 四、使用方法

## 1、引入框架

向工程中引入框架：CoreData.framework。

## 2、引入头文件，实现基类协议

引入头文件 #import "WASearchViewController.h"

并且继承类CWASearchViewController，实现它的协议MWAListCacheProtocal，MWALCCProtocal，示例如下：

U8CRM联系人ViewController

@interface CWALinkManViewController : CWASearchViewController

<MWAListCacheProtocal>

{

CWALinkManController \* iLinkManController;

}

U8CRM联系人Controller

@interface CWALinkManController : NSObject<MWALCCProtocal>

{

//viewcontroller

CWALinkManViewController \* iLinkManViewController;

}

## 3、初始化

在WALinkManViewController实现重载方法-(id)init;

如下：

//初始化

-(id)init

{

/\*

此处请参照CWASearchListViewStyle的定义，自行设置界面控件

特别注意，如果是多分组的，一定要设置iIsMutipleSection

aListViewStyle. iIsMutipleSection = YES;

\*/

CWASearchListViewStyle \* aListViewStyle =

[[[CWASearchListViewStyle alloc] initWithStyle:ELCSearchTSB] autorelease];

aListViewStyle.iIsMutipleSection = YES;

self = [super init:aListViewStyle];

if (self)

{

iLinkManController = [[CWALinkManController alloc] init];

iLinkManController.iLinkManViewController = self;

self.iMWAListCacheDelegate = self;

self.iMWALCCDelegate = iLinkManController;

iIsGetFirstData = NO;

}

return self;

}

### a 关键点：CWASearchListViewStyle自定义界面

目前可使用的如下的，及默认值，如下：

|  |  |
| --- | --- |
| BOOL iHaveTitleBar; //是否有titleBar,当没有titlebar时，left/right  //button,title都不会有 | YES |
| BOOL iHaveLeftButton; //有没有左边按钮 | YES |
| BOOL iHaveRightButton; //有没有右边按钮 | YES |
| BOOL iHaveTitleText; //有没有标题文本 | YES |
| BOOL iHaveSearchBar; //有没有搜索条 | YES |
| BOOL iHaveGroupButton; //有没有groupbutton | YES |
| BOOL iHaveRedLabel; //有没有红色的条 | NO |
| BOOL iHaveBottomArea; //有没有底部区域 | YES |
| BOOL iIsMutipleSection; //是否是多分组的形式 | NO |
| BOOL iIsCurViewSearch; //是否当前界面搜索，为NO，则不会弹出键盘 | YES |
| BOOL iIsCancelSQuit; //取消搜索按钮，是退出搜索界面功能 | NO |
| BOOL iIsFirstHistory; //是否默认显示历史记录，显示历史记录时 | NO |

对于CWASearchListViewStyle指定类型初始化，可选类型如下：

typedef enum

{

ELCSearchTSGB = 0,//含有标题，searcbar，groupbutton，底部区域

ELCSearchTSB, //含有标题，searcbar，底部区域

ELCSearchTSG, //含有标题，searcbar，groupbutton

}TWAListCacheSearchStyle;

其默认的属性值除了下面的不同外，其余都是上面表格相同

ELCSearchTSGB：和上面完全相同

ELCSearchTSB ：iHaveGroupButton = NO;

ELCSearchTSG ：iHaveBottomArea = NO;

Listcache内部根据这些标识来调整界面调整函数为：

/\*!

@method

@abstract 设置界面,根据TWASearchListStyle设置

\*/

-(void)setViewStyle;

## 4、实现列表的搜索，上拉刷新，下拉加载的数据请求

当初始化完成后，如果，产生了上拉刷新，下拉加载，搜索，点击历史记录的操作，基类需要请求数据时，就会通过代理通知子类实现，并返回。

### a 联网请求数据，基类通过以下通知子类请求数据

//没有groupbutton的情形

-(void)startSearch:(NSString\*)searchText

StartPos:(int)aStartPos

PageNum:(int)aPageNum;

//有groupbutton的情形

-(void)startSearchWithGroup:(NSString \*)searchText

StartPos:(int)aStartPos

PageNum:(int)aPageNum

GroupBtIndex:(int)aIndex

GroupBtTitle:(NSString\*)aBtTitle;

### b 子类请求数据，获取到数据后，返回给基类。

//是多分组的情形

先返回当前页，属于上一个分组的数目

-(BOOL)addLastGroupRowNum:(int)aGroupAddRowNum;

此处可以根据下列函数返回上一个分组的信息

/\*!

@method

@abstract 获取最后一个分组的信息

@discussion 只有当界面样式为多section时，显示

@result 如果当前界面时多section时，返回分组情况，单seciton返回nil

\*/

-(CWASearchListGroupDetail\*)getLastGroupInfo;

然后添加分组

-(BOOL)addNewGroup:(int)aGroupRowNum

WithGroupName:(NSString\*)aGroupName WithGroupKey:(NSString\*)aGroupKey;

CWASearchListGroupDetail 定义如下

@interface CWASearchListGroupDetail: NSObject

{

NSString \* iSectionName ; //分组名称

NSString \* iSectionKey ; //分组的key

int iSectionIndex ; //分组的位置，第几个分组

int iSectionRowNum; //该分组下面row的数目

}

注意：此处一定要将一次请求，返回的所有分组情况添加完毕后， 才能返回数据，注意，返回数据如果超过页数（目前是25）属于错误的情形。

添加分组完毕，返回数据

-(void)addSearchResultWithArray:(NSMutableArray\*)aResultArray;

//单分组的情形，直接返回数据

-(void)addSearchResultWithArray:(NSMutableArray\*)aResultArray;

## 5、初始化列表cell

基类无法统一所有的cell样式，所以，cell由子类自己初始化，实现以下三个委托函数即可

/\*!

@method

@abstract 初始化某一个cell的高度

\*/

- (CGFloat)setCellHeight:(UITableView \*)tableView

heightForRowAtIndexPath:(NSIndexPath \*)indexPath;

/\*!

@method

@abstract 初始化某一个cell,cell需要用户初始化

\*/

- (UITableViewCell \*)setCell:(UITableView \*)tableView

cellForRowAtIndexPath:(NSIndexPath \*)indexPath;

/\*!

method

@abstract 点击了tableView的一项

\*/

- (void)selectCell:(UITableView \*)tableView

didSelectRowAtIndexPath:(NSIndexPath \*)indexPath;

### a 关键点：cell的一些默认属性

|  |  |
| --- | --- |
| cell高度 | //cell的默认高度 |
|  | #define WA\_LISTCACHE\_CELLHEIGHT 70 |
| section高度 | //section的高度 |
|  | #define WA\_LISTCACHE\_SECHEIGHT 20.0f |
| section frame | //group head view 的fram |
|  | #define WA\_LISTCACHE\_HEADVIEWRECT CGRectMake(0, 0, 320, 20) |
| section 背景 | #define WA\_SEARCH\_SECTIONBG @"listsection\_bg.png" |

## 6、其他的一些委托方法

/\*!

@method

@abstract 删除搜索列表里面的一个元素

@param aIndexPath 删除元素的位置

@result 如果子类删除成功返回YES，如果子类操作失败，返回NO

\*/

-(BOOL)deleteASearchObject:(NSIndexPath\*)aIndexPath;

/\*!

@method

@abstract 左边按钮点击

\*/

-(void)leftBtClick;

/\*!

@method

@abstract 右边按钮点击

\*/

-(void)rightBtClick;

/\*!

@method

@abstract 开始编辑搜索条

\*/

-(void)startEditSearchBar;

/\*!

@method

@abstract 取消搜索界面

\*/

-(void)cancelSearchStatus;

# 五、基类的一些关键方法

/\*!

method

@abstract show spinner

\*/

- (void) showMySpinner;

/\*!

method

@abstract hide spinner

\*/

- (void) hideMySpinner;

/\*!

@method

@abstract 显示提示信息

@param aMsg 显示的信息

\*/

-(void)showMsg:(NSString\*)aMsg;

/\*!

@method

@abstract 请求接口出错。在搜索列表情况下子类请求数据失败，

应该调用父类的这个函数，以便，父类停止列表的刷新加载状态。

@discussion 该函数先hidde spinner,然后根据theError来判断是否需要显示

提示信息，如果theError为nil，不显示，如果theError不为nil，

则显示

\*/

-(void)requestFailedWithError:(NSError\*)theError;